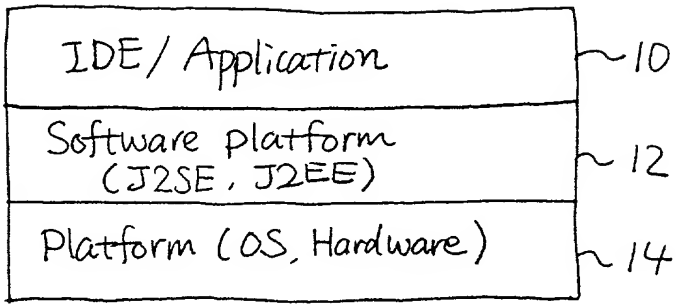

FIG. 1

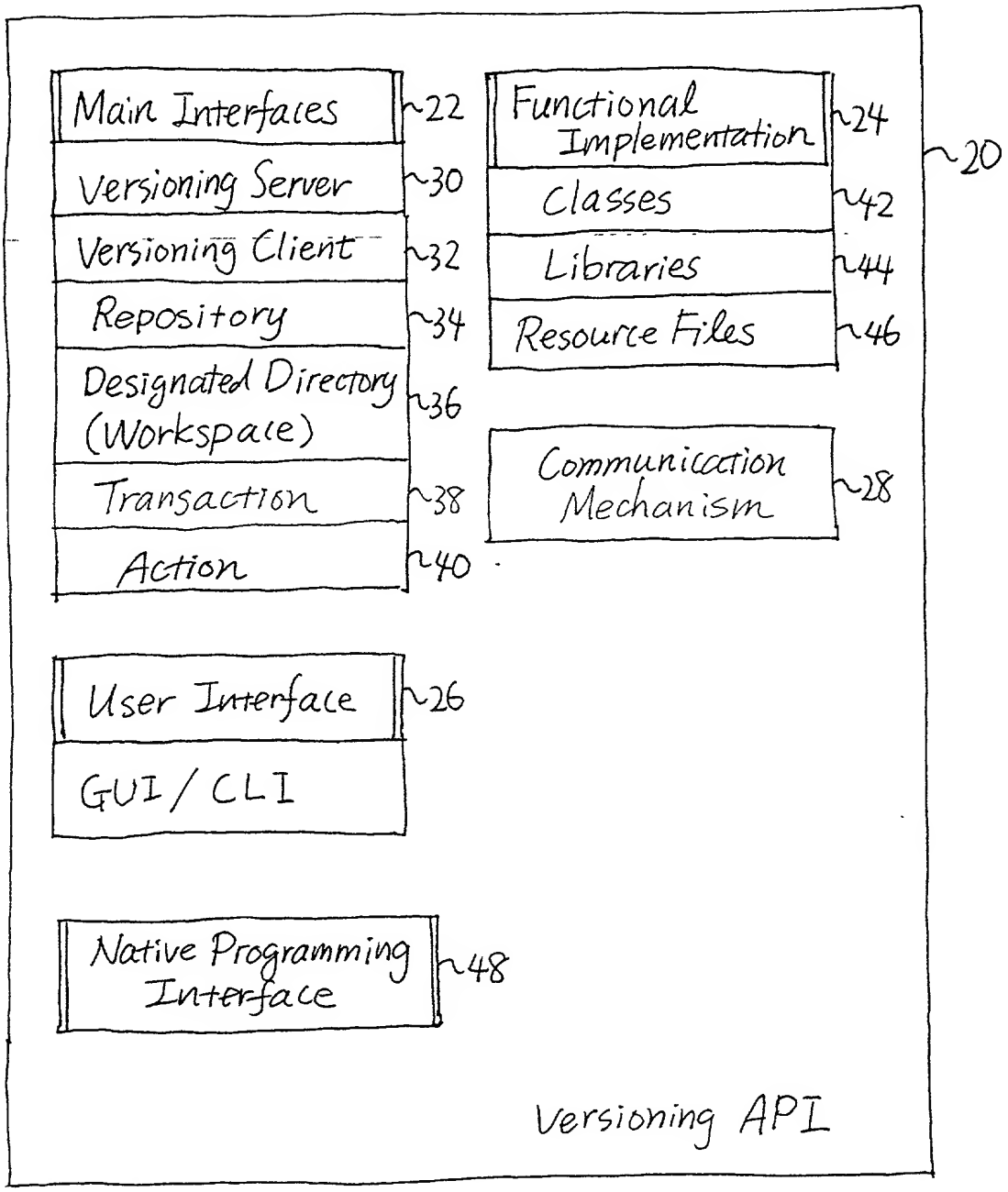

FIG. 2

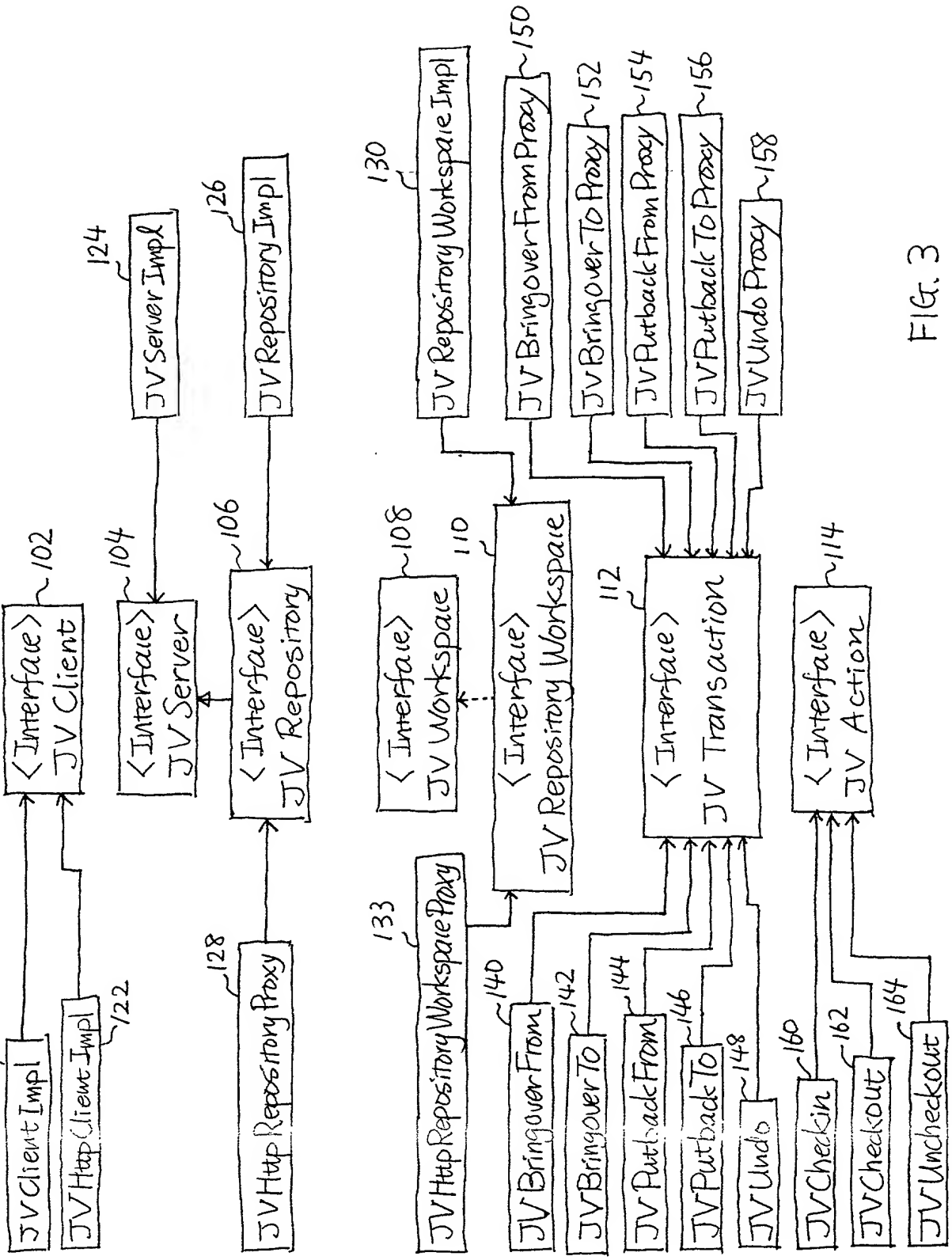

FIG. 3

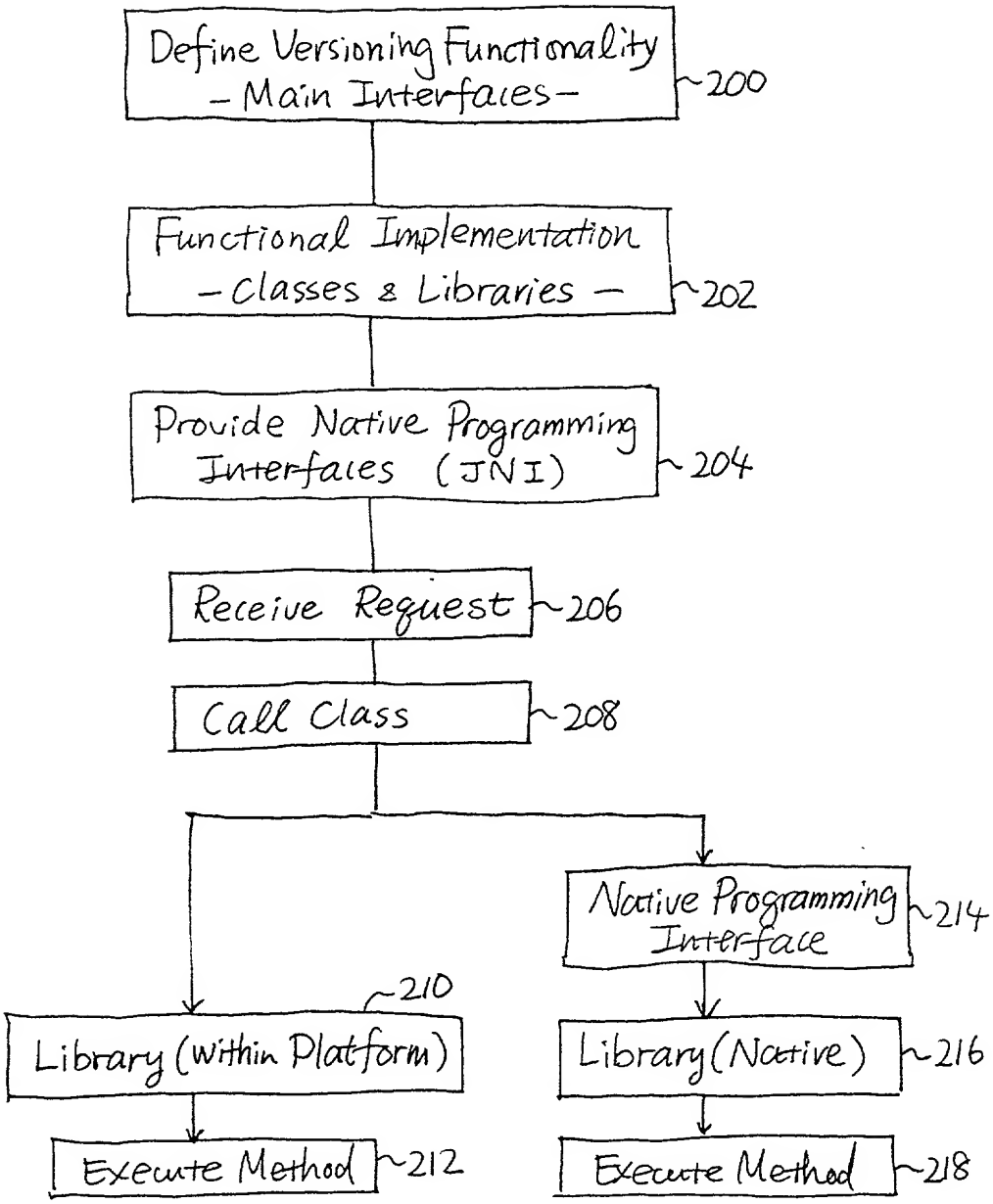

FIG. 4

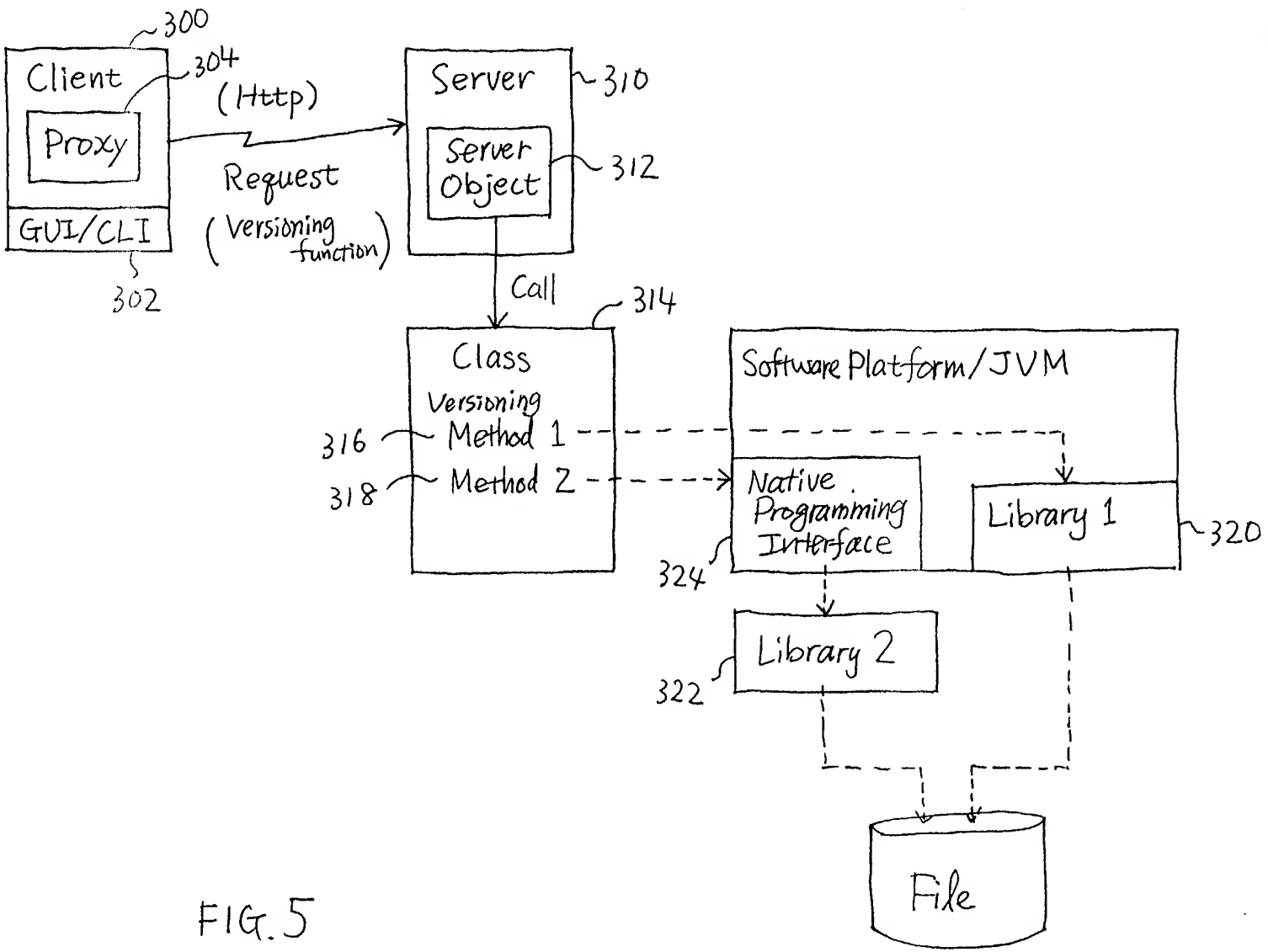
300
Client
304
Proxy
GUI/CLI
302
(Http)
Request
(Versioning function)
Server
310
Server Object
312
Call
314
Class
Versioning
316
Method 1
318
Method 2
Software Platform/JVM
Native Programming Interface
324
Library 1
320
Library 2
322
File

FIG. 5

```
package javax.version;

public interface JVServer {
public JVRepository logintoRepository(String username,
String
password, String repositoryname) throws JVException;
public void logoutofRepository(JVRepository repository)
throws
JVException;
}
```

FIG. 6

```
public interface JVClient implements JVServer
```

FIG. 7

```
package javax.version;

public interface JVRepository {

public JVRepositoryWorkspace
getRepositoryWorkspace(String
workspaceName) throws JVException;
public List listWorkspaces() throws TwException;
public void createWorkspace(String workspacename) throws
JVException;
public void deleteWorkspace(String workspacename) throws
JVException;
public void renameWorkspace(String oldWorkspacename,
String
newWorkspacename) throws JVException;
public void lock() throws JVException;
public void unlock() throws JVException;
}
```

FIG. 8

```
package javax.version;

public interface JVWorkspace {

public boolean equals(Object o);

/**
* Checks whether workspace exists
*/
public boolean exists() throws JVException;

/**
* Returns workspace abs name
*/
public String getName();

/**
* @return false if user does not have access for specified
operation
* @param operation is from the set of predefined constants
* This method should always keep track of the user.
*/
public bool checkAccess(String operation) throws JVException;

/**
* @return false if user does not have access for specified
operation
*/
public bool checkAccess(String operation, String user) throws
JVException;

/**
* Add children to workspace
*/
public void addChildren(List children) throws JVException;

/**
* Remove children from workspace
*/
public void removeChildren(List children) throws JVException;

/**
* Get children of workspace
* @return List of Strings
*/
public List getChildren() throws JVException;
```

FIG. 9A

```
/**
* Set children of workspace
* @param children List of Strings
*/
public void setChildren(List children) throws JVException;

/**
* Sets new parent workspace
*/
public void setParent(String newParent) throws JVException;

/**
* Gets new parent workspace as String
*/
public String getParent() throws JVException;

/**
* Sets new parent workspace
*/
public void setParentWorkspace(JVWorkspace newParentWorkspace)
throws JVException;

/**
* Gets new parent workspace object
*/
public JVWorkspace getParentWorkspace() throws JVException;

/**
* Changes workspace parent.
* Reparents the workspace and updates the old parent metadata.
*/
public void reparent(JVWorkspace newParent, boolean force)
throws
JVException;

/**
* Adds of files to the list of files in conflict.
* @param conflicts List of Strings
*/
public void addConflicts(List conflicts) throws JVException;

/**
* Removes files from the list of files in conflict
* @param conflicts List of Strings
*/
public void removeConflicts(List conflicts) throws JVException;
```

FIG. 9B

```
/**
* Sets the list of files in conflict.
* @param conflicts List of Strings
*/
public void setConflicts(List conflicts) throws JVException;

/**
* Gets the list of files in conflict.
* @return List of Strings
*/
public List getConflicts() throws JVException;

/**
* Locks workspace
* TwLock should have the fields: type, cmd, PID (SID), host,
* user, time. The constructor of JVLock should have these
arguments.
*/
public boolean lock(JVLock lock) throws JVException;

/**
* Returns list of workspace locks
*/
public List getLocks() throws JVException;

/**
* @return true if the lock exists
*/
public boolean checkLock(JVLock) throws JVException;

/**
* Unlocks workspace
*/
public boolean unlock(JVLock lock) throws JVException;

/**
* Creates a new physical ws, not a ws object
*/
public void createWorkspace() throws JVException;

/**
* Deletes a new physical ws, not a ws object
*/
public void deleteWorkspace(boolean metaonly) throws JVException;

/**
* Moves a workspace
*/
public void moveWorkspace(JVWorkspace dest) throws JVException;
```

FIG. 9C

```
/**
* Returns workspace description object
*/
public JVDescription getDescription() throws JVException;

/**
* Sets new description for workspace
*/
public void setDescription(JVDescription newDescription) throws
JVException;

/**
* Returns access control object
*/
public JVAccess getAccess() throws JVException;

/**
* Sets access control object
*/
public void setAccess(JVAccess newVal) throws JVException;

/**
* Gets history object
*/
public JVHistory getHistory() throws JVException;

public void setHistory(JVHistory newVal) throws JVException;
/**
* Gets putback validation object
*/
public JVPutbackValidation getPutbackValidation() throws
JVException;

/**
* Sets putback validation object
*/
public void setPutbackValidation(JVPutbackValidation newVal)
throws
JVException;

public JVStatus getWorkspaceStatus() throws JVException;
public OutputStream getFile(String relPathName) throws
JVException;
public void putFile(InputStream istream, String relPathName)
throws JVException;
}
```

FIG. 9D

```
package javax.version;

public interface JVTransaction {

public static final int NOT_STARTED = 0;
public static final int STARTED = 1;
public static final int FINISHED = 2;

/**
* stops the transaction
*/
public JVTransactionOutput getOutput() throws JVException;

/**
* @return one of the statuses defined in this interface
*/
public int getStatus() throws JVException;

/**
* @return a List of checksum (statetable) comparison of 2
workspaces
*/
public List init() throws JVException;

/**
* starts the transaction (non-blocking)
*/
public void start() throws JVException;

/**
* stops the transaction
*/
public void stop() throws JVException;
}
```

FIG. 10